A Knowledge Representation Approach to the Design of Architectures for Office Information Systems

Daniela MUSTO

Istituto di Elaborazione della Informazione
Consiglio Nazionale delle Ricerche
Via Santa Maria 46, I-56126 Pisa, Italy

DRAFT

Abstract

A framework for generating computer system Architectures that are suitable for realizing an Office Information System is presented. The Architecture generation process is seen as a mapping from a functional specification of the office expressed in terms of a conceptual scheme, and integrated with data concerning the existing office and users's needs, to a number of office Architectures, each satisfying by construction the given set of functional and non-functional requirements. Among such Architectures, the most appropriate one with respect to a cost-benefit trade-off is then selected, on the basis of performance measurement obtained via simulation. A methodology for achieving the mapping underlying the Architecture generation process is proposed, and a model for describing office Architectures is provided. Such a model relies on a knowledge base where hardware and software products are organized along with the knowledge for combining them into Architectures. A computer tool exists which supports an incremental construction of Architectures according to this model. The tool embodies the knowledge about hardware and software products, and about the possible ways of combining them into Architectures, in the form of a procedural semantic network, and makes accessible this knowledge to the Architecture designer through an interactive environment supported by a graphical interface.

Keywords: conceptual scheme, graphical interface, knowledge-based system, object-oriented languages, procedural semantic network, rule-based language, software development methodology.

1. Introduction

This paper describes a framework for generating computer system Architectures that are suitable for realizing an Office Information System (OIS).

The work presented here has been carried out within the ESPRIT Project No.813 *Tools for Designing Office Information Systems* (TODOS), which investigates methodological aspects of the OIS design. The purpose of the Project TODOS is the definition of a computer-based environment appropriate for supporting the design of OISs, where the design process is conceived as consisting of four strongly integrated phases [Pern86]: Requirements Collection and Analysis, Logical Design, Rapid Prototyping and Architecture Design.

The paper focuses on the Architecture Design phase of TODOS, which has the aim of identifying an Architecture appropriate for implementing the OIS being designed. Remark

that while the first three phases of TODOS are common to any classical software development methodology, the Architecture Design phase is a contribution of the Project to the area of OISs design. The presence of this phase, rather than focalizing the effort for realizing an OIS on the development of software [1], emphasizes the use of existing software and hardware products, towards the search for a configuration of them that defines the optimal office Architecture.

To achieve its aim, the Architecture Design phase is subdivided in two stages. The first stage, called Architecture Generation, produces a number of Architectures suitable for realizing the OIS, starting (a) from a functional specification of the office expressed in terms of a conceptual scheme, which is provided by the Logical Design phase, and (b) from a collection of data concerning the existing office and users' needs, which are provided by the Requirement Collection and Analysis phase through a requirements database [2] [Must89]. The second stage, called Architecture Selection, selects the most appropriate Architecture among those identified in the first stage, with respect to a cost-benefit trade-off and on the basis of performance measurements obtained via simulation [Dors89].

The work reported here concerns the Architecture Generation stage. The paper contains three main sections: Section 2 introduces a model for specifying the office Architecture; Section 3 describes an interactive computer tool which supports the incremental construction of the office Architectures according to that model; finally, Section 4 presents a methodology for mapping the collection of functional and non-functional requirements given as input to the Architecture Generation stage into a set of alternative office Architectures, each one of these satisfying by construction the input requirements.

2. A Model for Office Architectures

Modelling Architectures within the OIS design presents two basic requirements: on the one hand we must be able of representing real world Architectures whose functionalities can be matched with the architectural constraints stated implicitly in the conceptual scheme and either implicitly or explicitly in the requirements database; on the other hand, we must represent Architectures at the level of details suitable to specify the conditions of connectability and compatibility arising among architectural components, in order to be able of proposing *feasible* configurations of them, that is configurations that can be really used for implementing the OIS being designed.

Furthermore, office Architectures must be used as input to the Architecture Selection phase, thus the model selected for representing them must be structured in such a way that the translation of an office Architecture into a faithful queuing network model is natural and straightforward as possible.

Finally, the expansion properties [3] of the hardware products must be explicitly represented

[1] Thie development of software is the solution commonly adopted for realizing OISs. One examplification of this solution is given in [Paci85] and [Corre85].

[2] Remark that the conceptual scheme as well as the contents of the requirements database are verified through the Rapid Prototyping phase before becoming the input for the Architecture Generation stage, in order to be appropriately refined.

[3] The current technology has made the use of expansions very common in the practice of computer systems configuration. A given computer, for example, may be internally expanded by adding additional coprocessors, or internal memory, or interfaces; the same holds for peripherals. The computer may also be expanded by adding software packages, provided that these packages are compatible with the basic software equipment supplied by the manufacturer with the computer. Refer to [Cast88] for a more extended investigation of this subject.

in the office Architecture model, as they may influence the functionalities and/or the performance of the Architecture. This requires, on one side, to describe in the model the expansion properties of the commercial hardware products indicated by the manufacturer; and, on the other side, to specify in the model which expansions actually take place in the Architecture, if any, with respect to the each hardware occurrence.

In order to satisfy all the requirements indicated above, we have introduced the so called **Architecture Specification Model** (shortly, **ASM**). The major characteristics of ASM are presented below. For a complete description of ASM refer to [Cast88], while for a formalization of ASM through a first order theory [4] refer to [Cast89b].

An office Architecture is viewed in ASM as a set of interconnected hardware components that supports typical office activities. A hardware component is any machine that can be employed in the office and that is characterized by the presence of an electronic data exchange facility allowing for the integration of the machine in the office Architecture [5]; computers, input/output peripherals and storage peripherals are typical hardware components [6].

To be considered parts of the office Architecture, hardware components must be connected together trough local or remote connections. A local connection is realized by point-to-point connecting two hardware components through a dedicated physical link. A remote connection is realized by connecting as hosts the hardware components either to the same computer network or, alternatively, to different but communicating computer networks. Within the variety of existing computer networks [7], ASM only considers Local Area Networks (LANs), as they are the ones typically employed in the office.

The office Architecture supports the office activities by implementing the functionalities offered by the software packages running on the computer components. The selection of these packages in order to fit the requirements of the office activities is one of the crucial steps in the construction of Architectures. Another crucial step is that concerning the establishment of the local and the remote connections.

ASM specifies the office Architecture trough the following basic concepts: **Software Unit**, which represents any software package employed in the office; **Hardware Unit**, which represents any hardware product employed in the office; **Hardware-Component Unit**, which specializes the previous concept, and represents any hardware component [8] employed in the office, either in its basic version or in an expanded one (this last possibly including, in the case of Computer Units, also software expansions); **Subsystem**, which represents a set of Hardware-Component Units locally connected together, and is defined as a non empty set of Hardware-Component Units, comprising at least one Computer Unit, where the Hardware-Component Units are arbitrarily point-to-point connected between each other with the only restriction that no isolated Unit occurs; **Office Network**, which represents a set of Hardware-Component Units remotely connected together, and is defined as a non empty set of Subsystems interconnected through the same LAN.

---

[4] For the definition of a first order theory refer, e.g., to [Mend64].

[5] Remark that this requirement follows directly from the decision of modelling computer system Architectures.

[6] An example of a device not included among the hardware components because of the absence of an electronic data exchange facility is the copier machine.

[7] For an exemplification of this variety, refer, e.g., to [Stal84].

[8] Recall that by *hardware component* any machine equipped with an electronic data exchange facility is meant. The term *hardware product* is used, instead, to generically denotes any physical device employed in the office, and thus to indicate any hardware component as well as LANs, cables, expansion boards, etc..

Among the concepts introduced above, the first two are used to model the *constituents* of the office Architectures from the *commercial view-point.* The last three concepts, instead, are used to model the *constituents* of office Architectures

from the *structural view-point.* These last three concepts, in fact, define architectural components of increasing complexity; at the same time they delineate simple forms of office Architecture [9].

In its most general form, an office Architecture is seen in ASM as consisting of a non empty set of Office Networks, communicating between each other through common Subsystems. Thus ASM determines a top-down decomposition of the office Architecture trough the following abstraction levels: the **Architecture level**, where the office Architecture is defined in terms of Office Networks; the **Office Network level**, where the Office Networks are defined in terms of Subsystems and LANs; the **Subsystem level**, where the Subsystems are defined in terms of point-to-point connected Hardware Units; the **Unit level**, where the Units employed in the office Architecture are defined , and, in particular, the Hardware-Component Units are defined in terms of their internal parts [10] as well as of their possible expansions with respect to the basic version (with the inclusion of the specification of the software packages for the Computer Units).

The commercial hardware and software products whose instances can be employed in an office Architecture, are collected by ASM into an appropriate repository, called the **Catalogue**. Within the Catalogue the hardware and software products are grouped according to a functionality criterion, i.e. hardware (resp. software) products supporting (resp. providing) the same functionalities are arranged in the same class [11]. Thus software and hardware products are grouped into classes like , for the software products, the DBMSs, the graphical tools, the wordprocessings, and , for the hardware products, the computers, the storage peripherals, the LANs, and so on. The classes of hardware components so defined in the Catalogue are in turn subdivided into more specialistic classes, to take into account the substantial difference in the internal structure and/or in the performance of the machines belonging to the same class. These differences, in fact, represent very critical elements in the design of office Architectures. The Catalogue thus subdivides the computers into mainframes, minicomputers, personal computers, wordprocessors and integrated workstations; and, for example, subdivides the printers into impact printers, magnetic printer, laser printers, and so on.

Remark that the relationship between the commercial products in the Catalogue and the Units in the office Architecture resembles that between a prototype and its instances [Sowa84], and is clearly one-to-many. In fact many instances of the same commercial product may be employed in an office Architecture; each one of these instance is represented by a Unit, but all these Units refer to the same commercial product. Furthermore hardware instance of the same commercial hardware component can be expanded in different ways in the office Architecture.

[9] A Subsystem, for example, may be used to model a centralized solution for the office Architecture, like the one consisting of a set of terminals connected to a single mainframe.

[10] For a complete list of the internal parts of the hardware components that have been recognized as relevant in the modelling of office Architecture refer to [Cast88]. Among them we point out the *interfaces* and the *expansion slots*, that are used for modelling, respectively, the connectability and the expandability of the hardware components.

[11] Remark that other criteria could be be adopted for grouping the commercial products in the Catalogue. For example the one that emphasizes the manufacturer instead of the functionality. However we have adopted the functionality criterion because it seemed to us the most appropriate one for the consultation of the Catalogue during the construction of Architectures implementing the OIS.

We conclude the presentation of ASM by recalling that the expandability of the hardware components as well as their connectability into higher level architectural objects, like Subsystems and Office Networks, are controlled in ASM through the introduction of special relationships, called **compatibility relationships**, that arise among hardware components. These compatibility relationships must be consulted during the construction of the offices Architecture in order to guarantee its feasibility.

3. A Tool for Specifying Office Architectures

This section presents a tool that supports the specification of office Architectures according to ASM, the model for describing office Architectures introduced in the previous section.

As described above, ASM fundamentally embodies two different kinds of knowledge, that are orthogonal one with respect to the other. The first concerns the commercial hardware and software products that are available for the realization of the office Architecture, and is defined by the Catalogue; the other concerns the structure of the office Architecture, and is expressed in terms of Units, Subsystems and Office Networks.

The complexity of these two kinds of knowledge has suggested us to implement the tool as a knowledge-based system. The major characteristics of such a system, called **Architecture SPEcification System** (or, shortly, **ASPES**), are presented below. For a complete description of ASPES refer to [Cast88].

ASPES is a knowledge-based system that handles in an uniform way the different kinds of knowledge embodied in ASM. To this purpose it utilizes as knowledge representation language the language PSN [Leve79], which formalizes semantic network concepts within a procedural framework [12]. PSN provides the mechanisms for representing and manipulating objects and binary relationships between them, according to the modelling principles of object oriented languages, and it has been already employed in the development of knowledge-based systems [Mylo83].

The knowledge embodied in ASPES is made available to the Architecture designer trough an interactive environment supported by a graphical interface. The Architecture designer incrementally makes up the office Architecture being constructed by interacting with the ASPES interface. ASPES assists the Architecture Designer during the construction of the Architecture:

(a) by maintaining and making accessible a catalogue of available commercial hardware and software products that can be used in the definition of office Architectures;

(b) by providing a set of operations to incrementally build a description of an office Architecture;

(c) by guaranteeing that the resulting Architecture is a feasible one.

The architecture of ASPES is sketched in figure 1 and consists of two main modules: the **ASPES Knowledge Base** and the **ASPES user interface**.

The ASPES Knowledge Base (KB) is divided into three components: the **Kernel KB**, which contains the knowledge concerning the structure of the office Architecture; the **Catalogue KB**, containing the objects which represent the available commercial hardware

[12] Remark that the naturalness of semantic network formalisms in representing knowledge about domain of strongly interrelated objects, motivates the use of a formalism of such kind in coping with the Catalogue. The second kind of knowledge that must be handled by ASPES might have found in a rule-based language an appropriate representation scheme, as the R1 experience shows [McDe80]. However the tight interaction of these two kinds of knowledge in ASM has suggested us the use of PSN.

and software products and their compatibilities; and the **Architecture KB**, which includes the objects (i.e., Units, Subsystems, Office Networks and Architecture) constituting a particular Architecture. At the beginning of an ASPES session the Architecture KB is empty: it grows as the construction of an Architecture proceeds.

The **ASPES user interface**, which is a graphical interface, consists of two separated modules: the **Catalogue Maintenance Module**, which undertakes the acquisition of knowledge about hardware and software products, by providing operations for setting up and keeping updated the Catalogue, and the **Architecture Construction Module**, which allows to incrementally define Architectures, in a bottom-up fashion, by providing operations for querying the Catalogue and operations for querying and manipulating Architectures.

All the operations provided by the ASPES user interface are structured in a way that mirrors ASM, that is they are grouped by level, according to the type of objects that each operation processes, where the levels are those defined in ASM, including the Catalogue. At each level, there are operations for querying the system about the current status of the objects associated at that level, and operations for the manipulation (creation, removal, update) of such objects.

A prototypical implementation of ASPES is described in [Cast89b]. The implementation of ASPES has been carried out on a SUN 3/52 workstation. A version of PSN running on top of Franz Lisp has been used to implement the ASPES Knowledge Base. The ASPES User Interface has been implemented as a C program that runs under UNIX [13] and that uses the graphical facilities provided by the Sun View software library.

4. The Architecture Generation Process

The tool ASPES introduced in the previous section supports the Architecture generation process by guaranteeing the feasibility of the resulting Architectures, but not their matching with the OIS requirements. In fact the process of analysis and satisfactions of the OIS requirements is demanded integrally to the Architecture designer.

Our investigation of the Architecture generation process has enucleated some portions of the Architecture Generation stage that can be effectively realized by a tool, in collaboration with the Architecture designer, for supporting the analysis and satisfactions of the OIS requirements. These parts, typically, concern the control of some well-defined properties or the definition of default solutions.

Clearly, the ideal tool for supporting the Architecture Generation stage would be an expert system ables of integrally substituting the Architecture designer, and thus, in particular, ables of autonomously producing one or more architectural solutions satisfying all the OIS requirements that it receives as input. However this goal seems to us too ambitious to be accomplished, because of the inherent complexity of the domain under consideration, and of the difficulties encountered for acquiring and representing into a coherent formal framework the competence of the human expert.

We have individuated the implementable portions of the Architecture Generation stage trough the definition of a methodology, called **TODOS Mapping Methodology** (shortly, **TMM**) [Must89], which has the aim to mimic part of the steps followed by the Designer during his work.

[13] UNIX is a trademark of AT&T.

The Architecture generation process is seen by TMM as a mapping from a functional specification of the office expressed in terms of a conceptual scheme, and integrated with data concerning the existing office and users's needs which are collected within a requirements database, into a set of office Architectures, each satisfying by construction the given functional and non-functional requirements.

The functional requirements, in particular, are those specified by the conceptual scheme, and concern the form of the office activities and the data objects manipulated by them. The non-functional requirements are those expressed by the requirements database, and concern data like the periodicity of the office activities, the volume of the data involved by them, the location of the office workers carrying them out, the competence and the skill of these office workers, price constraints for the acquisition of new hardware or software equipment, constraints concerning the reutilization of the equipment already available in the office, etc..

TMM proposes a stepwise refinement process carrying out successive transformation of an office description preliminary derived from the conceptual scheme and the requirements database, and presenting the main functional and non-functional requirements of the OIS. The refinement process produces a list of office descriptions that realize different abstraction level views of the office, the last one of them providing the specification of the Architecture. Such descriptions are generated in a functionality preserving way and, at the same time, from the first to the last, with an increasing attention to the non-functional requirements of the OIS.

TMM consists of six steps, each one of these steps tackling one specific problem of the mapping and emphasizing the decisions involved by this problem. In particular, **Step 0** focuses on the **locations** of individual office workers in the office and on the **activities** performed by them; **Step 1** focuses on the **functionalities** supporting these activities; **Step 2** focuses on the definition of **Service Access Points** (shortly, **SAPs**) [Must88], that represent logical points in the office where subsets of the functionalities individuated at Step 1 are provided; **Step 3** focuses on the identification of **Subsystems** and **Office Networks**, which, according to ASM, partially fix the structure for the Architecture to be derived, and specifies them respectively as cluster of SAPS and as cluster of Subsystems; **Step 4** focuses on the specification of the **classes of hardware and software products** that must realize the Subsystems and the Office Networks; finally, **Step 5** focuses on the **office Architecture**, by constructing its specification with the support of ASPES.

We list below the basic properties that must be satisfied by a tool in order to implement TMM:

(a) every time a decision is required within the Architecture generation process, the tool asks the Architecture designer for the solution, but it also provides the designer with a set of default solutions that can be automatically realized, whenever selected;

(b) the consistency of the transformation of an office description into another one, within the sequence of steps defined by TMM, is automatically guaranteed by the tool;

(c) the tool supports backtracks points, allowing the Architecture designer to recover undesirable path during Architecture generation process;

(d) the tool supports the definition of different alternative solutions for each decision required within the steps of the Architecture generation process, in order to facilitate the Architecture designer in the structural comparison of parallel solutions;

(e) the tool is conceived as a *meta tool*: in fact it provides the Architecture designer with facilities for the utilization of other tools, like ASPES, for the specification of the Architecture, TQL [Fugi88], for the consultation of the conceptual scheme, and TST [Henr87],

for the consultation of the requirements database [14].

Due to the structure of the office descriptions individuated within TMM and to the nature of the tool ASPES, a tool implementing TMM can be realized trough a straightforward extension of the available tool ASPES. The results of a preliminary investigation on this direction are presented in [Must88]. Remark, in particular, that ASPES embodies technical solutions that directly inspire the realization of the properties (c), (d) and (a) for the target tool. Moreover the satisfaction of the properties (b) and (e) does not present conceptual difficulties within this framework. Thus the only crucial aspect in the realization of the tool implementing TMM is represented by the time factor, which probably will requires, in order to remain accettable, to distribute the code implementing the tool on different SUN workstations (by researving, for exmple, one to the handle of the graphical interface module, and another to the handle of the knowledge base module).

5. Concluding Remarks

In this paper we have presented a framework for generating computer system Architectures that are suitable for realizing an OIS. In particular, we have introduced a model for specifying office Architectures that relies on a knowledge base, where hardware and software products are organized along with the knowledge for combining these products into Architectures.

We have described a knowledge-based system (ASPES) that implements this model by using the knowledge representation language PSN [Leve75]. ASPES assists the Architecture designer during the construction of the Architecture by guaranteeing to him both the availability of an update catalogue of existing hardware and software products and the feasibility of the resulting Architecture, through an interactive environment supported by a graphical interface.

Finally we have proposed a methodology (TMM) for extending the automatical assistance provided by ASPES during the Architecture generation process also to those aspects concerning the satisfaction of the OIS requirements, so that the matching of the Architecture with such requirements is not integrally demanded to the Architecture designer.

A tool that implements the methodology can be realized trough a straightforward extension of the available tool ASPES. The results of a preliminary investigation on this direction are presented in [Must88].

Acknowledgments We acknowledge the contribution to this paper of all the TODOS partners, and particularly of the components of the Architecture Design workpackage, through discussion and common work. We are grateful to Aldo Pannocchia for having implemented the graphical interface of ASPES.

References

[**Barb87**] Barbic F., Fugini M.G., Maiocchi R., Pernici B., Rames J.R., Rolland C., *TODOS Conceptual Model and Specification Language*, Technical Report No. 2.2, ESPRIT Project TODOS, January 1987.

[**Bass87**] Bassanini G., Di Stefano F., Lunghi G., *TODOS Analysis Model Overview*, Technical Report No. 1.2.2.1, ESPRIT Project TODOS, December 1987.

[14] The tools TQL and TST have been developed within the Project TODOS to support, respectively, the Logical Design phase, and the requirements Collection and Analysis phase of the OIS design (see Section 1).

[**Cast88**] Castelli D., Meghini C., Musto D., *Architecture Specification Language: Design and Implementation*, Technical Report N0. 4.2, ESPRIT Project TODOS, June 1988.

[**Cast89a**] Castelli D., Meghini C., Musto D., *A Knowledge Representation Approach to Architecture Specification in the Office Information System Design*, B.Pernici and A.A. Verrijn-Stuart (Editors), Elsevier Science Publishers B.V. (North-Holland), IFIP 1989, pp.95-107.

[**Cast89b**] Castelli D., Meghini C., Musto D., *Architecture Specification in TODOS*, in TODOS Book, in preparation.

[**Corre85**] Correrini F., Ferretti V., Musto D., Pacini G., Schenone C., Turini F., Vantini G., *Manuale di riferimento del linguaggio COAL*, Quaderno CNET n.150, ETS:Pisa, 1985.

[**Dors89**] van Dorsselaer E., Heijmink F., *Architecture Simulation in TODOS*, in TODOS Book, in preparation.

[**Fugi88**] Fugini M.G., Maiocchi R., Pernici B., Pozzi, S., Stanga, M. and Zecca, U., *Specification Database Design: the C-TODOS tool*, Technical Report No. 2.3.1, ESPRIT Project TODOS, April 1988.

[**Henr87**] Henry P., *Overview of the TODOS Structuring Tool*, Technical Report No. 2.1.1, ESPRIT Project TODOS, September 1987.

[**Leve79**] Levesque H., Mylopoulos J., *A Procedural Semantics for Semantic Networks*, in Associative Networks, N.Findler (ed.), Academic Press, 1979.

[**McDe80**] McDermott J., *R1: A Rule-Based Configurer of Computer Systems*, CMU Tech. Rep, CMU-CS-80-119, Pittsburgh, April 1980.

[**Mend64**] Mendelson E., *Introduction to Mathematical Logic*, New York: Van Nostrand Company, 1964.

[**Mylo83**] Mylopoulos J., Shibahara T., Tsotos J.K., *Building Knowledge-Based Systems: The PSN Experience*, IEEE Computer, 16(10), October 1983.

[**Must88**] Musto D., *Evaluation of TODOS Conceptual Model and Study of Mapping Techniques*, Technical Report N0. 4.4, ESPRIT Project TODOS, December 1988.

[**Must89**] Musto D., *Architecture Specification in TODOS*, in TODOS Book, in preparation.

[**Pern86**] Pernici B., Vogel W., *An Integrated Approach to OIS Development*, ESPRIT Technical Week 86, Bruxelles, September 1986.

[**Paci85**] Pacini G., Turini F., *COAL: a design and implementation Language for Office Automation Systems*, Proceedings of the Conference *Distributed Systems on Local Network*, Pisa, June 24-28, 1985, ETS:Pisa, pp.473-505, 1985.

[**Sowa84**] Sowa J. F., *Conceptual Structures*, Addison-Wesley, 1984.

[**Stal84**] Stalling W., *Local Networks*, ACM Computing Surveys, 16 (1), March 1984.